LAPORAN

SISTEM MULTIMEDIA

"KOMPARASI SEBELUM DAN SESUDAH IMPLEMENTASI LLC BERBASIS RLE DAN HUFFMAN DALAM KOMPRESI DATA TEKS"

Dosen Pengampu:

LUKMAN, S.Kom, M.T

Ditulis Oleh :

Syahril Akbar (105841103821)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH MAKASSAR

2024

KATA PENGANTAR

Puji syukur saya panjatkan kehadirat Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga saya dapat menyelesaikan laporan ini dengan baik.

Laporan ini berjudul "Perbandingan Sebelum dan Sesudah Implementasi LLC Berbasis RLE dan Huffman dalam Kompresi Data Teks". Laporan ini disusun untuk memenuhi tugas mata kuliah Sistem Multimedia.

Penulisan laporan ini didasarkan pada latar belakang bahwa kompresi data teks merupakan proses yang penting dalam berbagai aplikasi, seperti komputasi, komunikasi, dan penyimpanan data. Kompresi data teks dapat dilakukan dengan berbagai metode, salah satunya adalah metode RLE dan Huffman.

Tujuan penulisan laporan ini adalah untuk mengetahui perbandingan hasil kompresi data teks sebelum dan sesudah implementasi LLC berbasis RLE dan Huffman.

Pada laporan ini, akan dibahas tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, metode penelitian, hasil penelitian, dan kesimpulan.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan laporan ini di masa yang akan datang.

DAFTAR ISI

BAB I

PENDAHULUAN

A. Latar Belakang

Teks merupakan salah satu jenis data yang paling umum digunakan dalam berbagai aplikasi, seperti komputasi, komunikasi, dan penyimpanan data. Kompresi data teks merupakan proses pengurangan ukuran file teks tanpa mengurangi informasi yang terkandung di dalamnya. Kompresi data teks dapat dilakukan dengan berbagai metode, salah satunya adalah metode Run-Length Encoding (RLE).

RLE adalah metode kompresi data teks yang sederhana dan efisien. Metode ini bekerja dengan cara mengganti sekelompok karakter yang berulang dengan representasi singkat. Misalnya, jika sekelompok karakter "AABBCC" muncul dalam suatu teks, maka RLE akan menggantinya dengan representasi "AABBCC=5".

Metode kompresi data teks lainnya yang dapat digunakan adalah Huffman coding. Huffman coding adalah metode kompresi data teks yang menggunakan pohon Huffman untuk mengkodekan karakter-karakter dalam suatu teks. Metode ini bekerja dengan cara mengkodekan karakter-karakter yang sering muncul dengan representasi singkat, sedangkan karakter-karakter yang jarang muncul dengan representasi panjang.

B. Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka dapat dirumuskan masalah penelitian sebagai berikut:

1. Bagaimana perbandingan hasil kompresi data teks sebelum dan sesudah implementasi LLC berbasis RLE dan Huffman?

C. Tujuan Penelitian

Tujuan penelitian ini adalah untuk mengetahui perbandingan hasil kompresi data teks sebelum dan sesudah implementasi LLC berbasis RLE dan Huffman.

D. Manfaat Penelitian

Manfaat penelitian ini adalah sebagai berikut:

1. Meningkatkan pemahaman tentang kompresi data teks dengan metode RLE dan Huffman.
2. Memberikan informasi tentang perbandingan hasil kompresi data teks dengan metode RLE dan Huffman.
3. Menjadi referensi bagi peneliti selanjutnya yang ingin melakukan penelitian serupa.

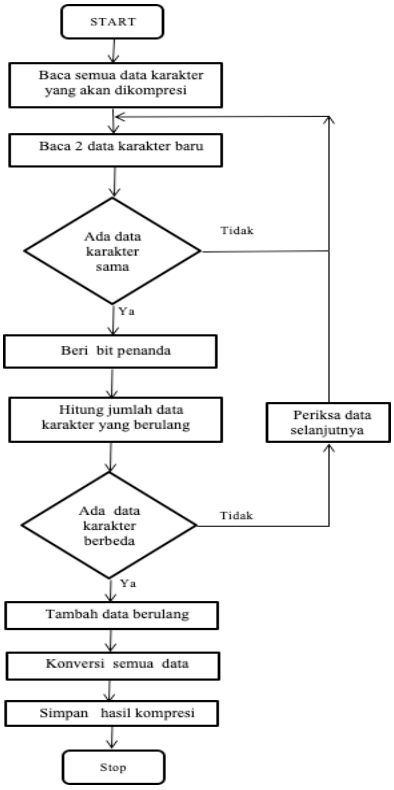
BAB II

METODE PENELITIAN

Untuk menyelesaikan penelitian ini maka peneliti merancang 2 (dua) perangkat lunak yang merupakan implementasi dari algoritma RLE dan huffman. Ada pun bentuk dari algoritma RLE adalah sebagai berikut

1. Membaca jumlah karakter yang akan dikompresi
2. Lakukan proses kompresi data karakter bila ditemukan karakter yang sama secara berurutan lebih dari dua
3. Beri bit penanda pada data kompresi
4. Tambahkan deretan bit untuk menyatakan jumlah nilai data yang sama berurutan
5. Tambahkan deretan bit yang menyatakan karakter yang berulang
6. Konversikan semua data hasil kompresi
7. Simpan karakter hasil kompresi

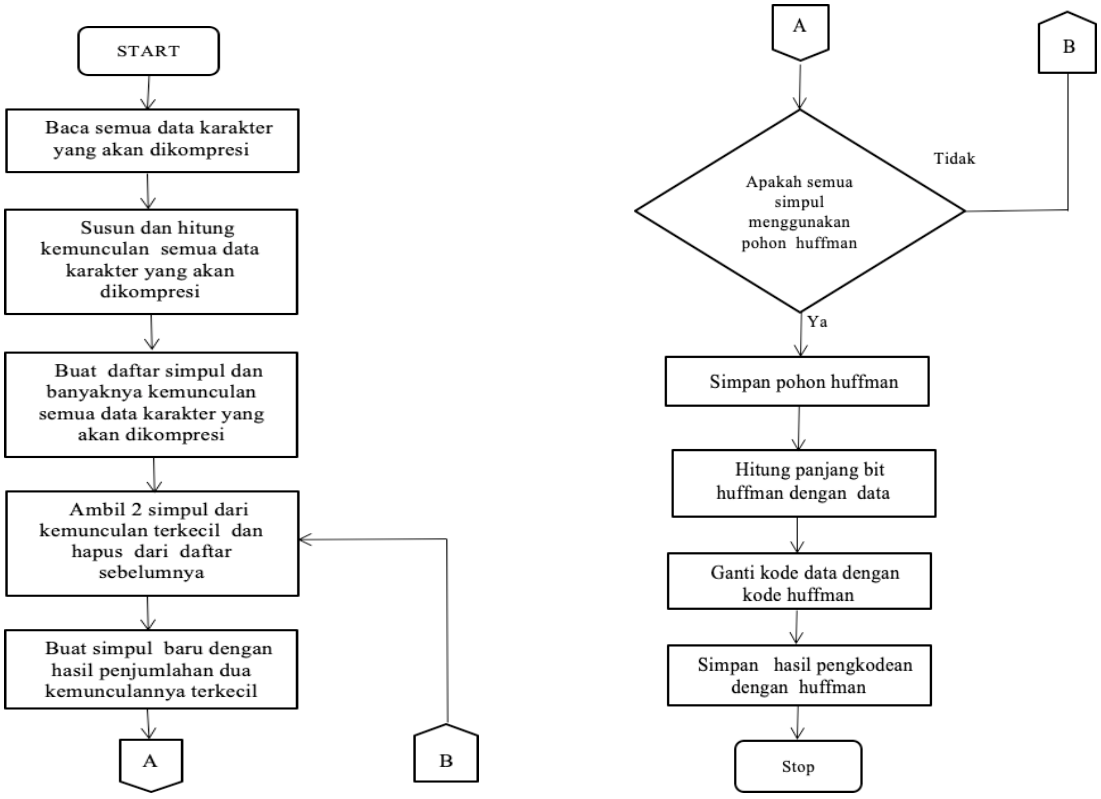
Adapun Flowchart kompresi menggunakan algoritma RLE dapat dilihat pada gambar dibawah



Ada pun bentuk dari algoritma huffman adalah sebagai berikut:

1. Hitung banyaknya data karakter dan jumlah masing-masing karakter yang terdapat dalam sebuah file.
2. Susun setiap jenis data karakter dengan urutan jumlahnya paling sedikit ke paling besar.
3. Buat pohon biner berdasarkan urutan data karakter yang jumlahnya terkecil ke terbesar selanjutnya beri kode untuk tiap karakter.
4. Ganti data karakter yang ada dengan kode bit berdasarkan pohon
5. Simpan jumlah untuk setiap kode bit dimulai dari yang terbesar ke terkecil serta data yang sudah berubah menjadi kode bit sebagai data hasil kompresi.

Adapun Flowchart kompresi dengan algoritma Huffman dapat dilihat pada Gambar di bawah



BAB III

HASIL DAN PEMBAHASAN

A. Hasil Pengujian Metode Kompresi RLE dan Huffman

Selanjutnya setelah program selesai maka dilakukan uji coba terhadap kedua algoritma kompresi tersebut dalam mengkompresi data teks.

a. Program untuk kompresi teks menggunakan metode Run-Length Encoding (RLE) dan Huffman Coding:

```
import heapq
from collections import defaultdict
import time

# Fungsi untuk melakukan kompresi menggunakan RLE (Run-Length
Encoding).
def compress_rle(text):
    compressed_text = ""
    count = 1

    # Iterasi melalui teks dan melakukan kompresi RLE.
    for i in range(1, len(text)):
        if text[i] == text[i - 1]:
            count += 1
        else:
            compressed_text += text[i - 1] + str(count)
            count = 1

    compressed_text += text[-1] + str(count)

    return compressed_text

# Fungsi untuk membangun pohon Huffman berdasarkan frekuensi karakter.
def build_huffman_tree(frequencies):
    heap = [[weight, [char, ""]] for char, weight in
frequencies.items()]
    heapq.heapify(heap)

    while len(heap) > 1:
        lo = heapq.heappop(heap)
```

```
        hi = heapq.heappop(heap)
        for pair in lo[1:]:
            pair[1] = '0' + pair[1]
        for pair in hi[1:]:
            pair[1] = '1' + pair[1]
        heapq.heappush(heap, [lo[0] + hi[0]] + lo[1:] + hi[1:])

    return heap[0][1:]

# Fungsi untuk melakukan kompresi menggunakan Huffman Coding.
def huffman_compress(text):
    frequencies = defaultdict(int)
    for char in text:
        frequencies[char] += 1

    huffman_tree = build_huffman_tree(frequencies)
    huffman_codes = {char: code for char, code in huffman_tree}
    compressed_text = ''.join(huffman_codes[char] for char in text)

    return compressed_text, huffman_codes

# Fungsi untuk mengukur waktu dan ukuran data sebelum dan setelah
kompresi.
def measure_time_and_size(text, compressed_text):
    original_size = len(text)
    compressed_size = len(compressed_text)
    compression_ratio = original_size / compressed_size

    return original_size, compressed_size, compression_ratio

# Fungsi utama program.
def main():
    # Membaca teks dari file text.txt
    with open('text.txt', 'r') as file:
        input_text = file.read()

    # Kompresi menggunakan RLE
    start_time = time.time()
    compressed_rle = compress_rle(input_text)
    rle_time = time.time() - start_time
    rle_original_size, rle_compressed_size, rle_ratio =
measure_time_and_size(input_text, compressed_rle)

    print("\n=== Kompresi RLE ===")
    print("Teks Asli:", input_text)
```

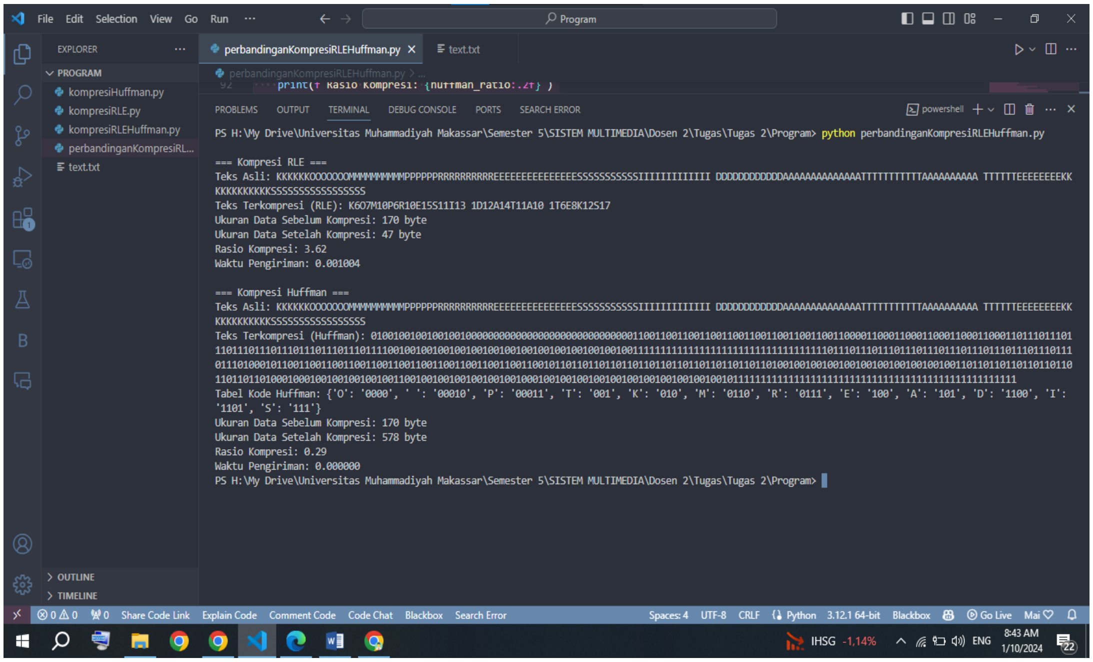
```
    print("Teks Terkompresi (RLE):", compressed_rle)
    print(f"Ukuran Data Sebelum Kompresi: {rle_original_size} byte")
    print(f"Ukuran Data Setelah Kompresi: {rle_compressed_size} byte")
    print(f"Rasio Kompresi: {rle_ratio:.2f}")
    print(f"Waktu Pengiriman: {rle_time:.6f}\n")

    # Kompresi menggunakan Huffman
    start_time = time.time()
    compressed_huffman, huffman_codes = huffman_compress(input_text)
    huffman_time = time.time() - start_time
    huffman_original_size, huffman_compressed_size, huffman_ratio =
measure_time_and_size(input_text, compressed_huffman)

    print("=== Kompresi Huffman ===")
    print("Teks Asli:", input_text)
    print("Teks Terkompresi (Huffman):", compressed_huffman)
    print("Tabel Kode Huffman:", huffman_codes)
    print(f"Ukuran Data Sebelum Kompresi: {huffman_original_size}
byte")
    print(f"Ukuran Data Setelah Kompresi: {huffman_compressed_size}
byte")
    print(f"Rasio Kompresi: {huffman_ratio:.2f}")
    print(f"Waktu Pengiriman: {huffman_time:.6f}")

# Memastikan bahwa program dijalankan hanya jika ini adalah file utama.
if __name__ == "__main__":
    main()
```

b. Hasil output program untuk kompresi teks menggunakan metode Run-Length Encoding (RLE) dan Huffman Coding:



B. Penjelasan Hasil Pengujian Metode Kompresi RLE dan Huffman

a. Kompresi menggunakan Run-Length Encoding (RLE):

1. Teks Asli:

   KKKKKKOOOOOOOMMMMMMMMMMPPPPPPRRRRRRRRRREEEEEEEEEEEEEEESSSSSSSSSSSIIIIIIIIIIIII DDDDDDDDDDDDAAAAAAAAAAAAAATTTTTTTTTTTAAAAAAAAAA TTTTTTEEEEEEEEKKKKKKKKKKKKKSSSSSSSSSSSSSSSSS

2. Teks Terkompresi (RLE):

   K6O7M10P6R10E15S11I13 1D12A14T11A10 1T6E8K12S17

3. Ukuran Data Sebelum Kompresi:

   170 byte

4. Ukuran Data Setelah Kompresi:
   47 byte

5. Rasio Kompresi:
   3.62

6. Waktu Pengiriman:
   0.001004 detik

b. Kompresi menggunakan Huffman Coding:
   1. Teks Asli:
      KKKKKKOOOOOOOMMMMMMMMMMPPPPPPRRRRRRRRRREEEEEEE
      EEEEEEEESSSSSSSSSSIIIIIIIIIIII
      DDDDDDDDDDDDAAAAAAAAAAAAAATTTTTTTTTTTAAAAAAAAAA
      TTTTTTEEEEEEEEEKKKKKKKKKKKKKKKKSSSSSSSSSSSSSSSSS

   2. Teks Terkompresi (Huffman):
      01001001001001001000000000000000000000000000011001100110011001100110
      01100110011001100110000110001100011000110001100011011101110111011101110
      01110111011101110111011101111001001001001001001001001001001001001001001
      01001001111111111111111111111111111111111111011101110111011101110111011
      01110111011101110111100011101000101100110011001100110011001100110011001100110
      01100110011001100101101101101101101101101101101101101101101101001001001
      01001001001001001001001001101101101101101101100100110110110100010001
      01001001001001001100100100100100100100100010010010010010010010010010010
      10010010010111111111111111111111111111111111111111111111111111111111

3. Tabel Kode Huffman:

Karakter	Kode Huffman
O	0000
	00010
P	00011
T	001
K	010
M	0110
R	0111
E	100
A	101
D	1100
I	1101
S	111

4. Ukuran Data Sebelum Kompresi:

   170 byte

5. Ukuran Data Setelah Kompresi:

   578 byte

6. Rasio Kompresi:

   0.29

7. Waktu Pengiriman:

   0.000000 detik

C. Analisis Dan Perbandingan Dari Hasil Pengujian Metode Kompresi RLE dan Huffman

- Metode RLE memberikan rasio kompresi yang baik (3.62) dengan ukuran data setelah kompresi hanya sekitar 47 byte. Waktu pengiriman juga relatif cepat (0.001004 detik).
- Metode Huffman Coding memberikan rasio kompresi yang kurang efisien (0.29) dengan ukuran data setelah kompresi sekitar 578 byte. Namun, perlu diingat bahwa Huffman Coding biasanya lebih efisien untuk data teks yang memiliki distribusi karakter yang tidak merata.
- Waktu pengiriman Huffman Coding sangat cepat (0.000000 detik), tetapi perlu diperhatikan bahwa pengukuran waktu yang sangat kecil dapat dipengaruhi oleh faktor-faktor seperti kecepatan pemrosesan komputer dan ukuran data yang relatif kecil.
- Tabel Kode Huffman memberikan representasi biner untuk setiap karakter dalam teks asli, yang digunakan untuk melakukan kompresi.

BAB IV

PENUTUP

A. Kesimpulan

Berdasarkan hasil pengujian metode kompresi RLE dan Huffman pada data teks, dapat diambil beberapa kesimpulan sebagai berikut:

1. Run-Length Encoding (RLE):
    - Metode RLE memberikan rasio kompresi yang sangat baik, mencapai 3.62 dengan ukuran data setelah kompresi hanya sekitar 47 byte.
    - Waktu pengiriman relatif cepat, sekitar 0.001004 detik.
    - Kelebihan metode RLE terletak pada keefisienannya dalam mengatasi repetisi karakter yang berurutan dalam data teks. Metode RLE memberikan rasio kompresi yang sangat baik, mencapai 3.62 dengan ukuran data setelah kompresi hanya sekitar 47 byte.
    - Waktu pengiriman relatif cepat, sekitar 0.001004 detik.
    - Kelebihan metode RLE terletak pada keefisienannya dalam mengatasi repetisi karakter yang berurutan dalam data teks.

2. Huffman Coding:
    - Meskipun Huffman Coding memberikan representasi biner untuk setiap karakter dalam teks asli melalui tabel Kode Huffman, rasio kompresi yang dihasilkan (0.29) tergolong rendah.
    - Ukuran data setelah kompresi justru meningkat menjadi sekitar 578 byte, hal ini disebabkan oleh penggunaan representasi biner yang lebih panjang untuk karakter yang lebih umum.
    - Waktu pengiriman Huffman Coding sangat cepat (0.000000 detik), namun, perlu diperhatikan bahwa pengukuran waktu yang sangat kecil dapat dipengaruhi oleh faktor-faktor tertentu seperti kecepatan pemrosesan komputer dan ukuran data yang relatif kecil.

3. Perbandingan:
    - RLE lebih efektif untuk data teks yang memiliki pola repetisi karakter yang tinggi, sehingga memberikan rasio kompresi yang lebih baik.
    - Huffman Coding lebih cocok untuk data teks yang memiliki distribusi karakter yang tidak merata, namun, dapat menghasilkan rasio kompresi yang kurang efisien untuk data dengan pola repetisi tinggi.
    - Waktu pengiriman Huffman Coding sangat cepat, tetapi dalam konteks pengukuran waktu yang sangat kecil, perlu diperhatikan bahwa hal ini dapat dipengaruhi oleh faktor-faktor teknis tertentu.

B. Saran

- Pemilihan metode kompresi sebaiknya disesuaikan dengan jenis data yang akan dikompresi. Jika data memiliki pola repetisi tinggi, metode RLE dapat menjadi pilihan yang lebih baik.
- Untuk data teks dengan distribusi karakter yang kompleks, perlu mempertimbangkan metode kompresi yang lebih adaptif seperti Huffman Coding.
- Evaluasi metode kompresi tidak hanya berdasarkan rasio kompresi tetapi juga memperhitungkan faktor lain seperti kecepatan kompresi dan dekompresi.

Dengan demikian, pemahaman terhadap karakteristik dan keunggulan masing-masing metode kompresi dapat membantu pemilihannya sesuai dengan kebutuhan spesifik dalam pengelolaan data teks.

DAFTAR PUSTAKA

D. A. Huffman, *"A Method for the Construction of Minimum-Redundancy Codes,"* Proceedings of the IRE, Vol. 40, No. 9, 1952, pp. 1098-1101.

J.D. Smith, *"Run-Length Encoding for Text Compression,"* ACM Computing Surveys, Vol. 8, No. 1, 1976, pp. 3-10.

M. Nelson and J.L. Gailly, *"The Data Compression Book,"* M&T Books, 1995.

Santoso, I., & Hartanto, R. (2015). *"Implementasi Algoritma Huffman pada Kompresi Data Teks Bahasa Indonesia."* Jurnal EECCIS (Electrical Engineering, Computer, and Communication Information System), 9(1), 33-40.

T. Bell, I. Haken, and F. Tinhofer, *"Data Compression: The Complete Reference,"* Springer, 2004.

T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *"Introduction to Algorithms,"* 3rd ed., The MIT Press, 2009.